Distributed Reliable Computing

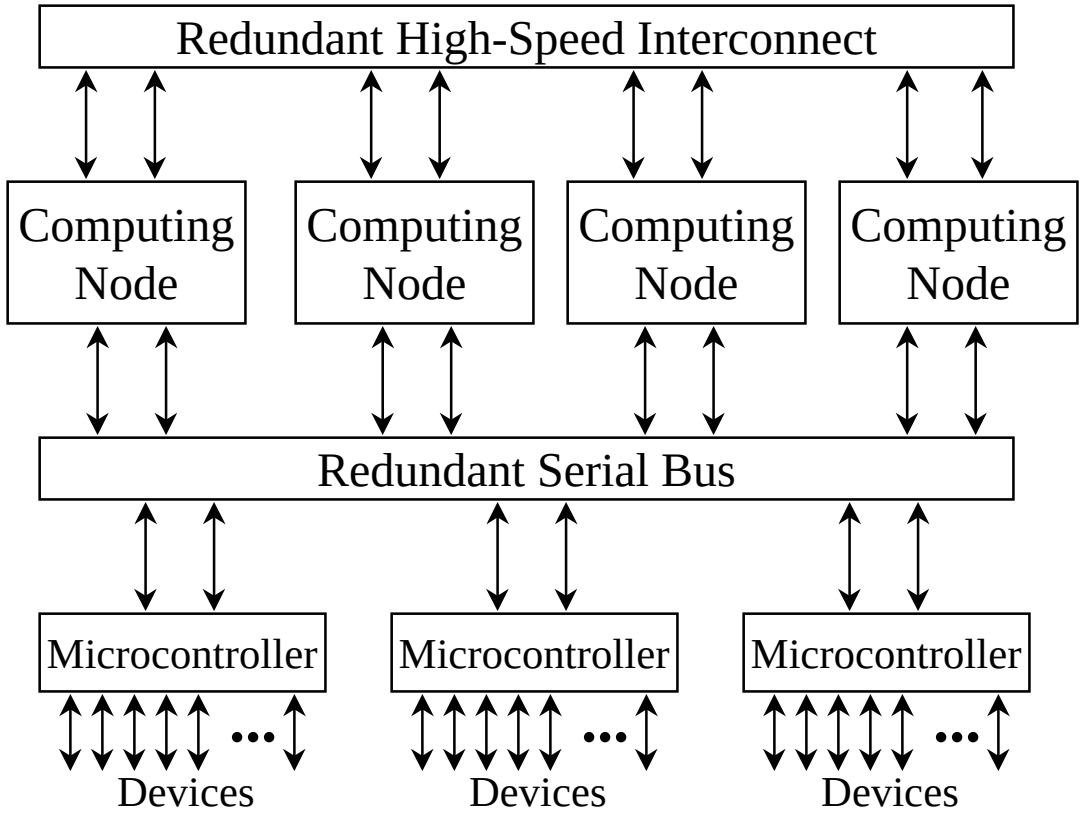
John Beahan

2nd DARPA Fault-Tolerant Computing Workshop
Jet Propulsion Laboratory
September 10-11, 1997

SIFT: Reliable Computation in Software At High Fault Rates

- Focused on achieving adequate reliability in the presence of previously unheard-of levels of radiation-induced faults
- Fault detection coverage is crucial:
    - Application reliability is challenging, but even more,
    - System coordination and fault management software must be highly resistant to internal inconsistency in order to avoid major system disruption
- Two main approaches:
    - Object-based paradigm for higher-reliability applications, i.e. main avionics:
      - Few processors, moderate performance, relatively high reliability
      - Configurable replication, voting, consensus, state management,
      - Emphasis on achieving “fail-notify” semantics by multiple techniques
    - “Cradle-to-grave” algorithmic methodology and techniques for high-performance, highly parallel applications:
      - Enforce consistency from moment data arrives at instrument
      - High-efficiency, tunable fault detection

Radiation Susceptibility Assumptions and Strategy



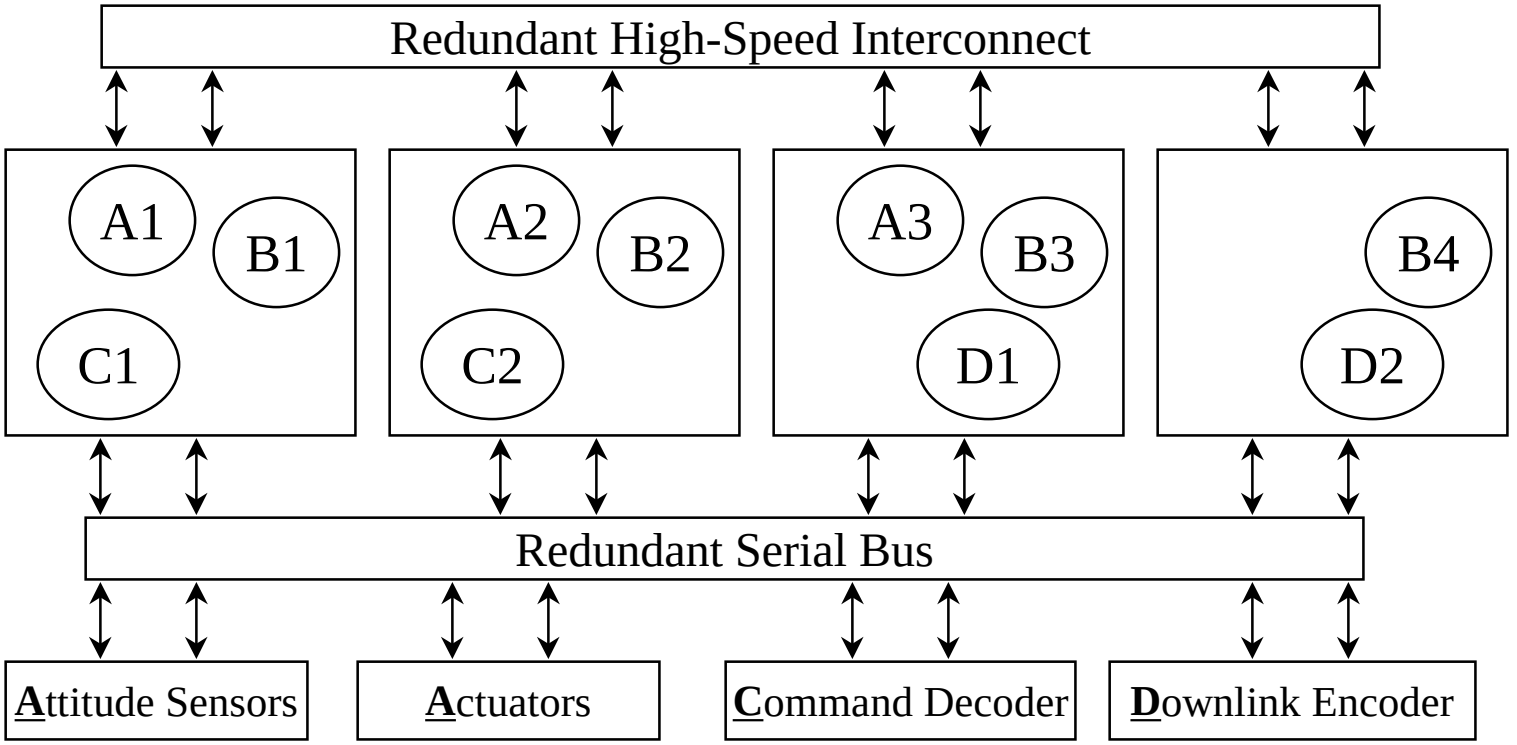
Rad-"firm" interconnect and computing nodes: no destructive latchup; infrequent Single-Event Latchup requiring power cycling

Rad-Hard Serial Bus

Rad-Resistant Microcontrollers, via either redundancy or rad-hard devices

Rad-Hard Lowest-level Device Controller Electronics

High-Reliability Scenario: Primary Avionics



- Control and I/O tasks replicated according to criticality, e.g., A = Attitude Control, B = Bus Interface, C = Command Processing, D = Downlink Telemetry
- Device microcontrollers implement simple protocols to cross-check inputs from multiple replicated tasks, e.g. checksums or voting in hardware/software
- Mastership of I/O operations at replica task level, allows parallel processing; master does data transfer, subordinate(s) transfer checksums to/from microcontrollers
- Failure model is at computing node level, possibly affecting multiple masters

Object-Based Reliability Services

Model

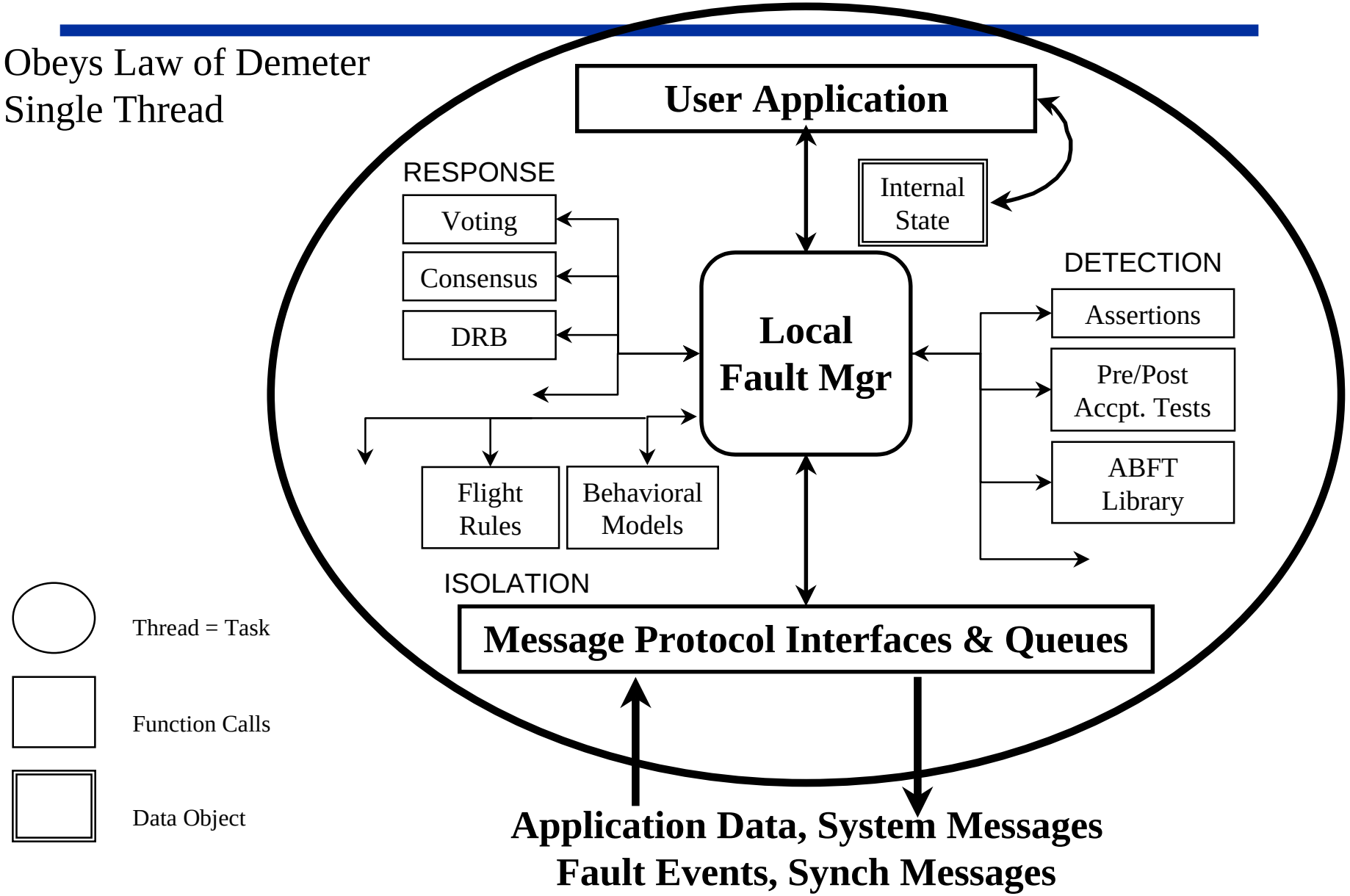
- Coarse-grained super-objects, single thread, each contains one user application object
- Message passing between super-objects
- User application objects may contain finer-grained objects, communicating via function calls
- Fault tolerance services introduced via:
    - Message transport layer and interfaces: replication, voting, consensus
    - Object-level FDIR libraries
    - Node-level coordination of activity on self and other nodes -- no central master node
- 2 classes of objects:
    - Those which obey the Law of Demeter and use orderless messages
    - All others, e.g. MPI/PVM task
- Until recently concentrated mainly on the LoD type, for reliable general-purpose avionics, e.g. commanding, telemetry, attitude control, etc.

Rationale

- The addition of task replication and interprocessor communication to the MPF task/object model increases the pressure to minimize the number of tasks (threads)

Reliable Task Object

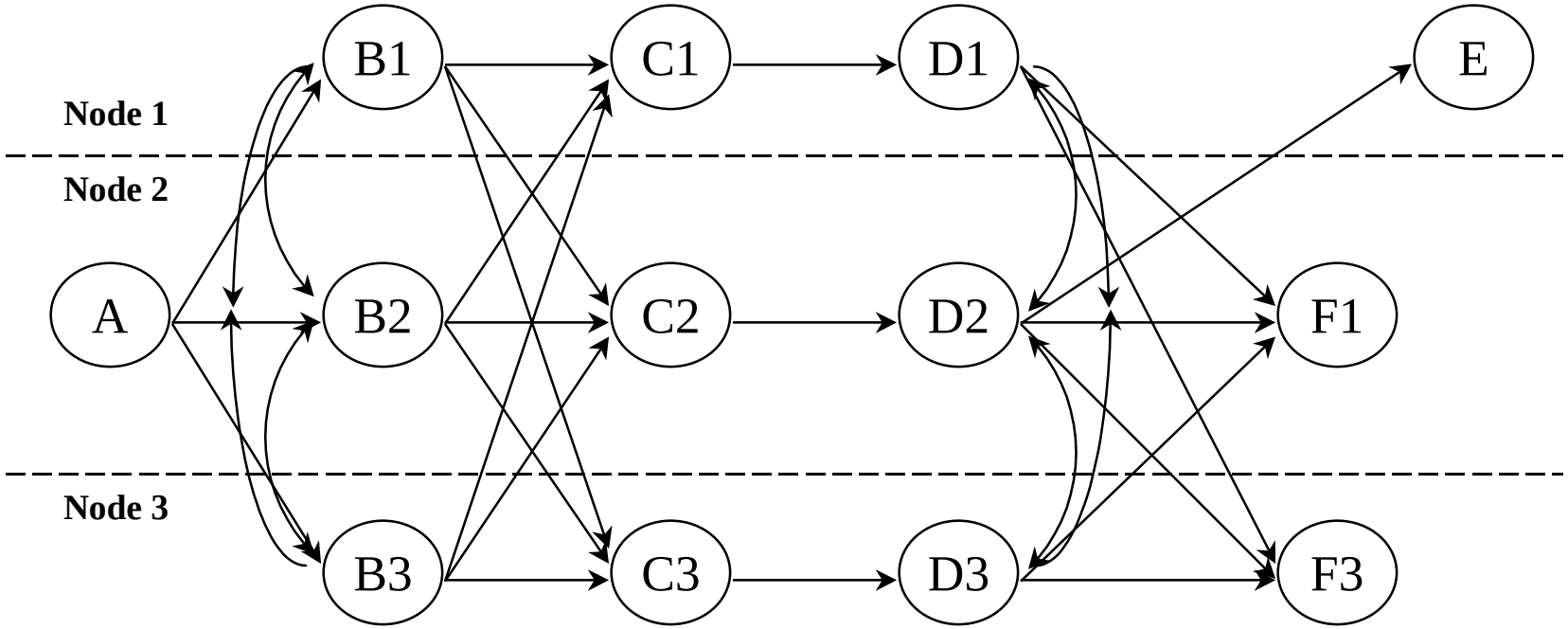
Obeys Law of Demeter
Single Thread



Configurable Replication and Communication

Replica behaviors and communication options:

- Input consensus (B's)
- Voting (B's to C's and D's to F's): exact match, data averaging, etc.
- Point-to-point (C's to D's) for increased performance
- Output consensus (D's)
- Single output by dynamically elected replica master (D's to E)



- Plan for erroneous critical data
- "Harden" the software protocols used to coordinate activities and perform fault tolerance functions: time replication, software-based basic block signatures, tamper-proofing
- Make the infrastructure as brittle (fault sensitive) as possible
- Understand and tailor the low-level fault propagation modes of the interconnect
  - hardware mitigation: prevent host memory corruption by disallowing interface bus master
  - software mitigation: alternative device driver formulations

Collaboration with Flight System Testbed

- Extensive facilities for integrated hardware/software development
- Existing expertise, and a suite of advanced integrated flight software prototypes
- Law of Demeter tasking and communication model -- single thread, event delivery with timeout functionality
- Real-time hardware emulating flight instruments
- TRAMEL messaging layer:
  - Message broadcast (not guaranteed)
  - Supports multiple protocols simultaneously, e.g. UDP, VxWorks queues
  - Multiplatform
  - Support for protocol-based management of resources: memory pools, allocators
  - Propagation of configuration information about executing tasks (threads)

Summary and Status

Status

- Initial prototype of object-based SIFT system operational and integrated with Flight System Testbed's Advanced Flight Software prototype
- Ad hoc group communication protocols
- Quality of service model currently beneath message layer, above OS
- Current implementation on 68040's communicating via UDP on a VME bus

Plans

- Migrate to PowerPC platform linked with redundant Myrinets
- Key issue is integrating QoS into kernel
- Emphasize development of efficient fault detection methods